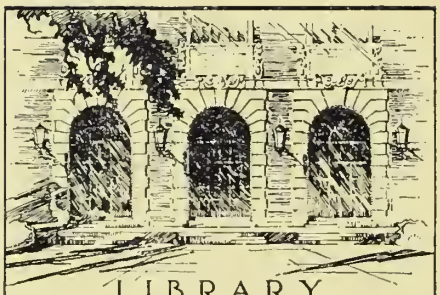


DIGITAL COMPUTER LABORATORY

UNIVERSITY OF ILLINOIS

URBANA, ILLINOIS

REPORT NO. 161

DESIGN OF AN ITERATIVE PATTERN RECOGNITION PROCESSOR*

by

S. R. Ray† and K. C. Smith†

April 22, 1964

This work is supported in part by

Contract No. AT(11-1)-1018 of the Atomic Energy Commission

ACKNOWLEDGMENTS

We wish to acknowledge an indebtedness to the originator and instigator of the ILLIAC III project, Professor B. H. McCormick, and to Professor R. Narasimhan under whose guidance the study of iterative processing methods have proceeded.

The aid of Messrs. Dennis Hall, Donald Dodson, Vincent Lum, and William Mayberry is also gratefully acknowledged.

SUMMARY

The logical design of the iterative processor of the ILLIAC III computer is described from an engineering viewpoint. The Iterative Array, a 1024-bit modular parallel processor, and the Transfer Memory, a 48 × 1024-bit orthogonal read/write memory, are treated. Illustrative examples are presented which relate various elements of the hardware to its uses.

This paper extends and deepens the system description of the Pattern Articulation Unit, discussed by McCormick.

SUMMARY

The logical design of the iterative processor of the ILLIAC III computer is described from an engineering viewpoint. The Iterative Array, a 1024-bit modular parallel processor, and the Transfer Memory, a 48 × 1024-bit orthogonal read/write memory, are treated. Illustrative examples are presented which relate various elements of the hardware to its uses.

This paper extends and deepens the system description of the Pattern Articulation Unit, discussed by McCormick.

1. INTRODUCTION

This paper discusses the engineering design of the iterative portion of a visual pattern recognition computer: ILLIAC III. General features and raison d'etre of the ILLIAC III have been treated previously [1].

It is convenient to view the ILLIAC III system as comprising two semi-autonomous processors: (1) a contemporary processor for arithmetic operations; and (2) an iterative processor for nonarithmetic, planar (i.e., two-dimensional) word operations. These two processors share a small matrix memory called the Transfer Memory. The arithmetic processor communicates to this memory in 48 bit words while the planar processor or "Iterative Array" communicates to the orthogonal memory axis in 1024 bit (32 × 32) words. The Iterative Array-Transfer Memory, together with their control unit, constitute an iterative computer.

The Iterative Array plus Transfer Memory may equally well be described as a content-addressable memory as seen from the arithmetic computer. Indeed, the concepts of the iterative processor [2-4] and of the content addressed memory [5-9], originally considered as disparate notions, converge in practice to synonymity. It will be convenient to employ the terminology developed in content-addressed memory work to describe engineering features and illustrate basic uses of the iterative processor.

The equipment which is considered here has been designed for visual pattern processing--specifically, the automatic scanning, measuring, and data reduction of bubble-chamber photographs. The engineering design incorporates the results of these simulation studies, including the concomitant development of nonarithmetic reduction methods for an iterative organization that has proven to be a practical, economical, balanced design [10].

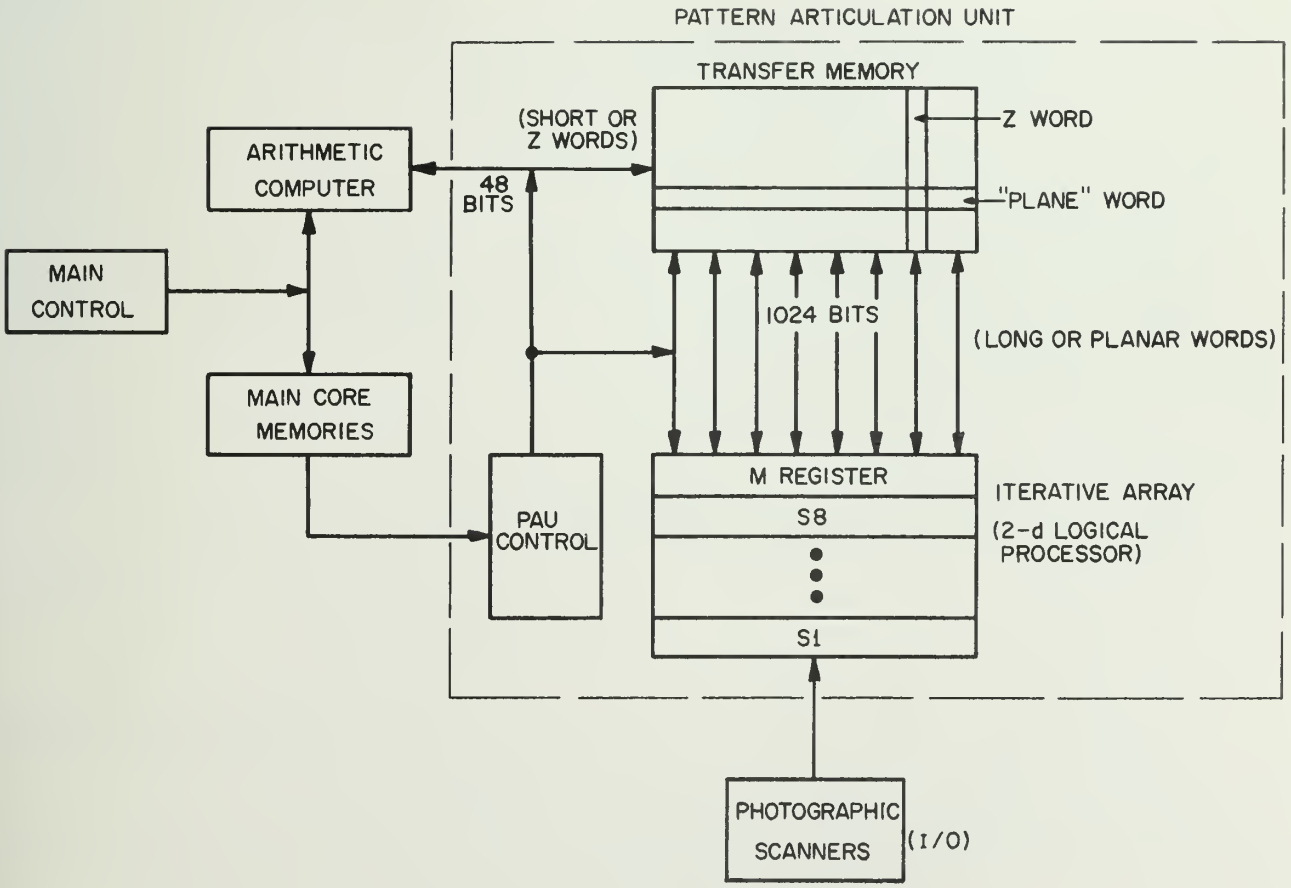
A major effect of this orientation is seen in the Iterative Array, which provides only very fundamental logical operations in exchange for a rich and flexibly interconnected environment. Similarly, reflecting the relatively high ratio of Iterative Array to Transfer Memory operations, the inclusion of nondestructive read facilities in the latter could not be economically justified.

The system organization will be briefly described in Sec. 2, the Transfer Memory discussed in Sec. 3. The Iterative Array is then considered in Sec. 4. The last section, Sec. 5, is devoted to examples of the use of various parts of the equipment.

2. THE ILLIAC III SYSTEM

A highly simplified diagram of the ILLIAC III system is given in Fig. 1. Short or "Z" words (48 bits) may be transferred in either direction between the Arithmetic Computer and Transfer Memory (TM). Long or planar words (1024 bits) are transferred between the TM and Iterative Array (IA) by way of the M register. Eight other long registers, S1-S8, exist in the IA. Each of the 1024 modular sections of the IA, containing one bit of each of the nine fast registers, is identical and all are identically controlled. The modular sections are arranged in a two-dimensional square array of size 32 × 32 with a border of overflow bits.

In content-addressed memory terminology, the M register is comparable to the interrogation response register. The encoding and marking operations required to retrieve the results of nonunique interrogation responses are also required in the present equipment but for a wider range of uses. A broader discussion of the system is available [1].



ILLIAC III SYSTEM
FIG. 1

3. THE TRANSFER MEMORY

The Transfer Memory (Fig. 2) is a random access, linear select core memory of 48 × 1024 bits. Considered as a two-dimensional matrix, the memory is arranged for read or write operations either by rows or by columns. An encoder is also included in the Transfer Memory.

The 1024 bit words, lying in the plane orthogonal to the Z axis, are called "plane" or "long" words while the 48 bit words are "Z" or "short" words (aligned in the Z direction).

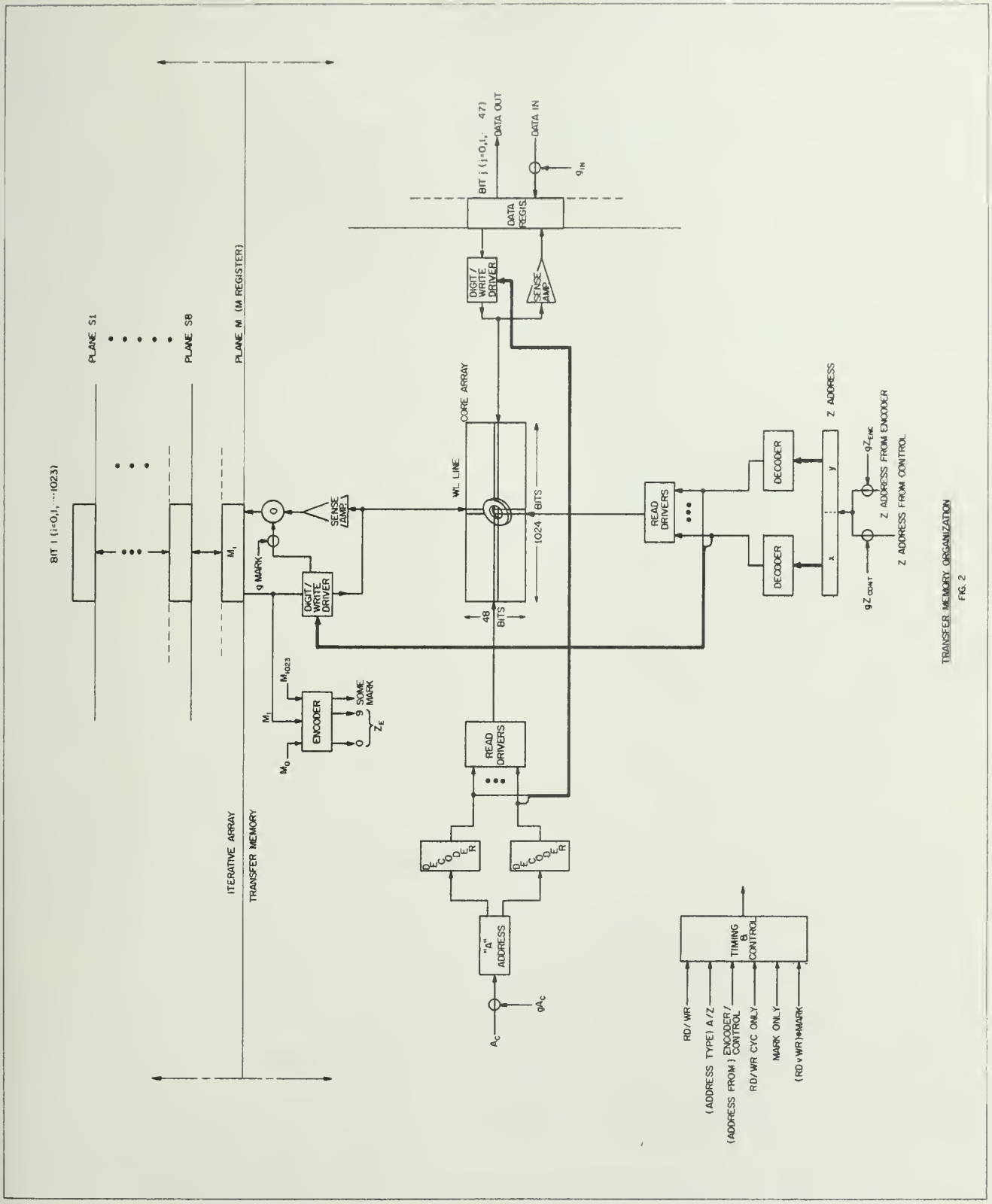
Access to both word types is performed in a 2 μsec destructive mode cycle. A one-core-per-bit, linear selection organization is used. The cores are 29 × 19 × 6.5 mil ferrite switched to 80 per cent or more of the available flux. Read drives of 350 ma-t (corresponding to 200 nsec switching time) are applied by sink-source switches to one-turn windings. The total write mmf is 240 ma-t developed by one-turn windings in the x,y direction carrying 120 ma and by two-turn windings in the Z direction which carry 60 ma.

Access to Z words is straightforward, but the planar word access requires some elaboration.

Plane Word Operations

Plane words are treated at the circuitry level as eight 128-bit segments which are read or written simultaneously. The M register of the Iterative Array acts as the data register for plane words.

There are two a priori requirements which effectively shape the detailed organization of this section of the memory. One requirement is that there exist the possibility of setting "1" (or "marking") into any specified unique bit of the 1024-bit M register, a MARK operation. A full decoding of a 10-bit address is needed for this since the address occasionally originates from the control unit.



TRANSFER MEMORY ORGANIZATION

FIG. 2

The second requirement, dictated by the need for storing plane words, is simply that there be (at least) one digit driver for each bit of the plane words, i.e., 1024 digit drivers.

These two requirements plus the generation of write current for Z words can be economically mated as described in the sequel.

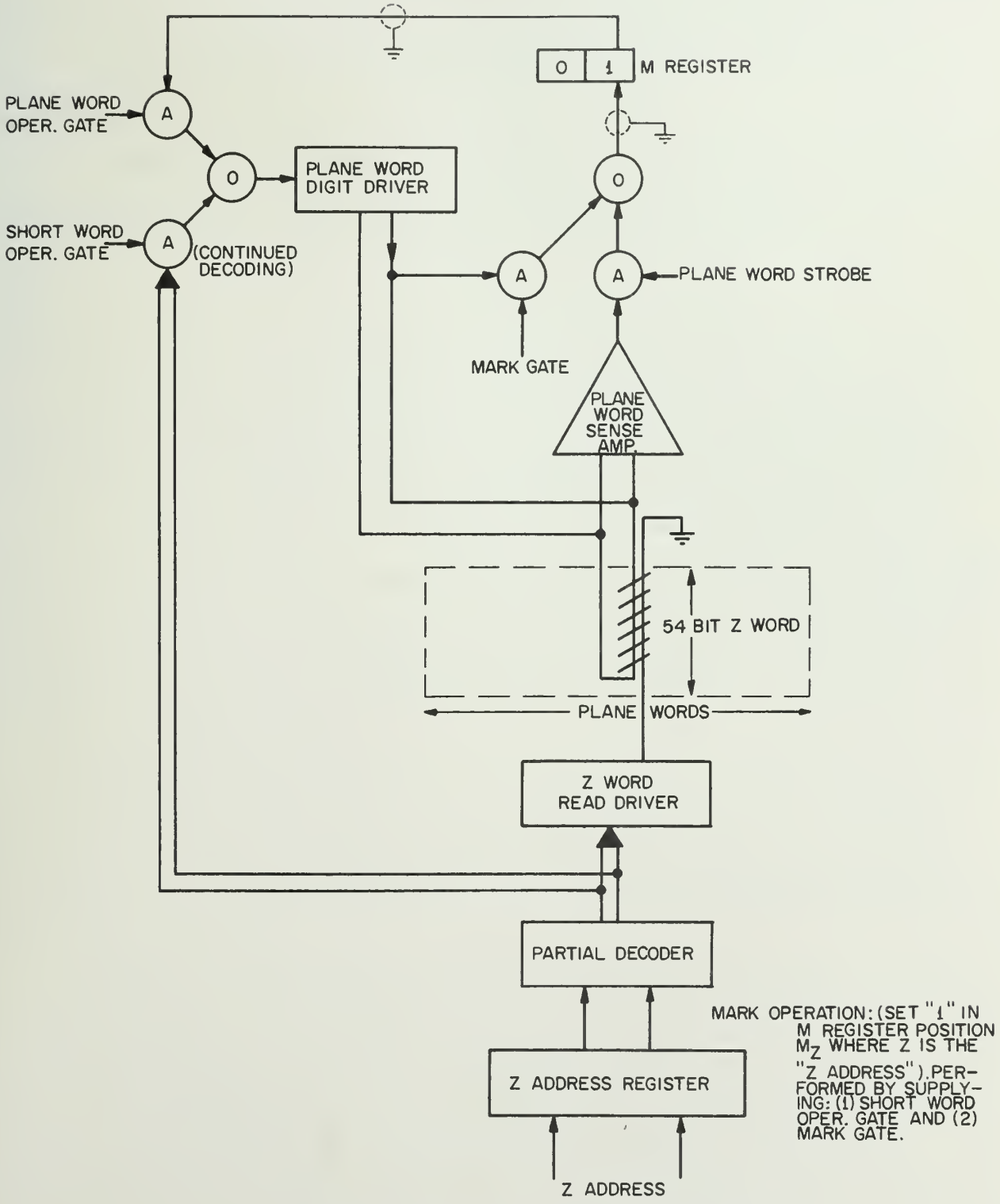
Referring to Fig. 3, it can be seen that the Plane Word Digit Driver may be activated by the M register (containing the information to be written) during the writing phase of a plane-word operation cycle. In this case, the digit driver acts in the role which its name implies.

A Z-word operation cycle may selectively activate one and only one of the 1024 Plane Word Digit Drivers, specified by the 10-bit Z address, whereupon the driver produces Z word line write current. Moreover, the "digit driver" output may also be gated (by the Mark Gate) to mark the M register which is not directly engaged, of course, in the principal data manipulations for a Z word cycle. Thus, MARK operations may be selectively included in Z-word read or write cycles at no cost in time beyond that of performing the Z word operation alone.

In summary, the costs of the large quantity of Plane Word Digit Drivers are amortized over three uses: as plane word digit drivers, as Z word write drivers, and as MARK signal generators with only relatively small costs in diodes and logic drivers required to select the various modes.

The MARK operation may also be performed independently of storage operations by supplying Short Word Operation Gate and Mark Gate signals after having placed the necessary address in the Z register. In this case, the digit driver which is activated causes no irreversible core flux change since no other drivers are simultaneously active.

Advantage is also taken of the shortness of digit lines (WL lines) which link only 48 cores,‡ and allow the use of a two-turn winding, thereby halving the digit current and doubling the sense signal amplitude. The length



EQUIPMENT FOR PLANE WORD OPERATIONS AND MARK

FIG. 3

also makes it possible to control the digit current simply by series resistance and driver saturation. It should also be noted that the cores are operated with equal digit and write mmf's (120 ma-t) so that no change is required in the current-setting circuits when the driver changes from digit driver to write driver status.

It was not found possible, unfortunately, to reduce greatly the cost of the sense amplifiers either directly or by multiple use tactics.

Encoder

The M register, which serves dually as the plane word data register of the Transfer Memory and as a register of the Iterative Array, delivers inputs to the encoder. The encoder consists of combinational circuitry which delivers (1) a 10-bit binary number (Z address) equal to the minimum coordinate or bit number of the M register which contains "0" and (2) a one-bit signal ("SOME MARK") indicating the existence of at least one "0" in the M register. Zeros may be interpreted as interrogation responses, that is, those words for which an associative criterion is true.

The encoded address of a "0" in M is developed and sent to the Z Address Register of the TM, where it may be used to erase (reset to "1") the corresponding zero by means of the MARK operation, thus allowing the encoder to proceed to the next zero.

A similar procedure has been proposed [6, 11] for resolving nonunique interrogation responses in associative memory designs. It should be noted, however, that the address of a mark ("response") is more frequently important here than are the contents of a responding memory location. Thus, it is of paramount importance to encode addresses to the maximum entropy form (10 bits, here), a premise which has some effect upon the choice of encoder design, and is at variance with the usual requirements for content addressed memory.

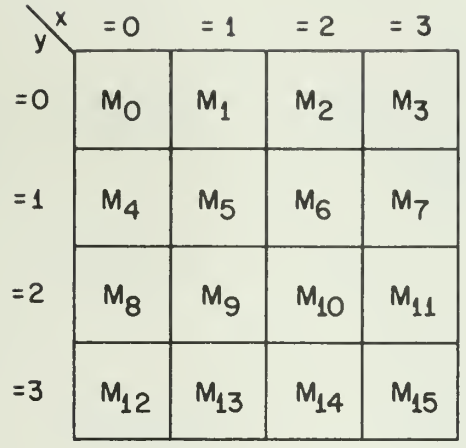
Although the encoder design is not a problem of major proportions, it is a recurring one. It is of some interest, perhaps, that a design which is is at once fast and economical can be described by a simple heuristic.

The encoding strategy is conveniently described by conceptually arranging the encoder inputs, in a two-dimensional matrix. Assume the 10-bit encoder output, Z = (y,x) where the ordered pair, y and x, are the most and least significant 5-bit groups of Z, respectively, and the numerical relation Z = 32y + x holds. The minimum Z such that M_Z = 0 is desired. If the set of values of (y,x) represent the row and column coordinates of the matrix in the usual way (see Fig. 4), then the minimum numerical value of Z for which a zero occurs in the matrix is found by first selecting the uppermost row containing at least one zero and, in that row, the leftmost column containing zero. The row and column thus selected clearly establish the minimum value of Z for which a zero exists.

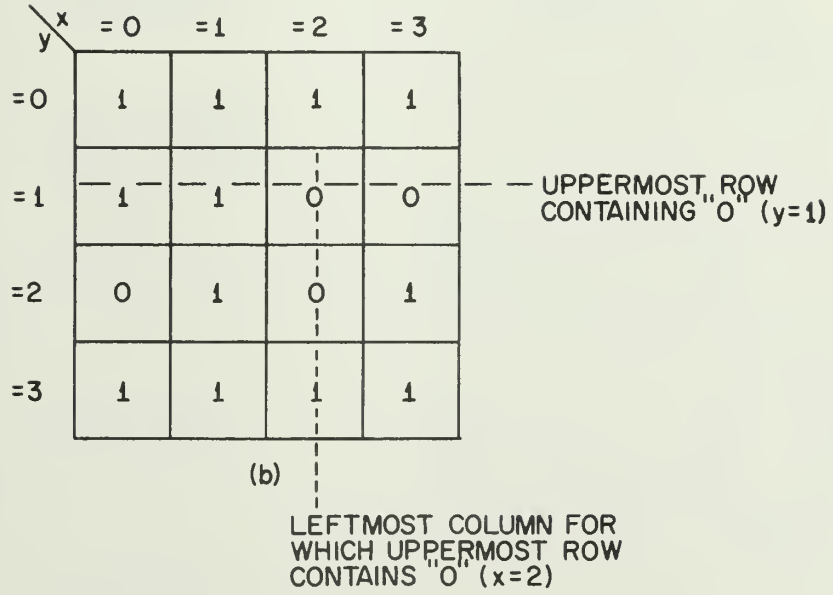
The logical realization of this procedure for a 1024 input encoder is shown in Fig. 5.

The 32-input NOR circuits correspond to the rows and columns of the matrix of Fig. 4. The priority circuit for the Y group (rows) effectively selects the uppermost row containing zero, delivering y_i = 1 for that row (row i) and forcing all other y signals to zero. The selected y_i then gates the 32 M signals, corresponding to entries in its row, into a similar priority circuit where the desired x is formed. Subsequent encoding of the Z address and generation of the SOME MARK signal is obvious.

The priority chain can be segmented for faster operation by sending advanced inhibit signals to indicate to lower rows, that some higher priority row has already been selected. In the limit of minimum segment length (one y group), the encoded address can be attained in 11 circuit operation times[§] with fanout from the inhibit drivers being 31 (= $\sqrt{n}$ - 1 where n is the number of bits).



(a)



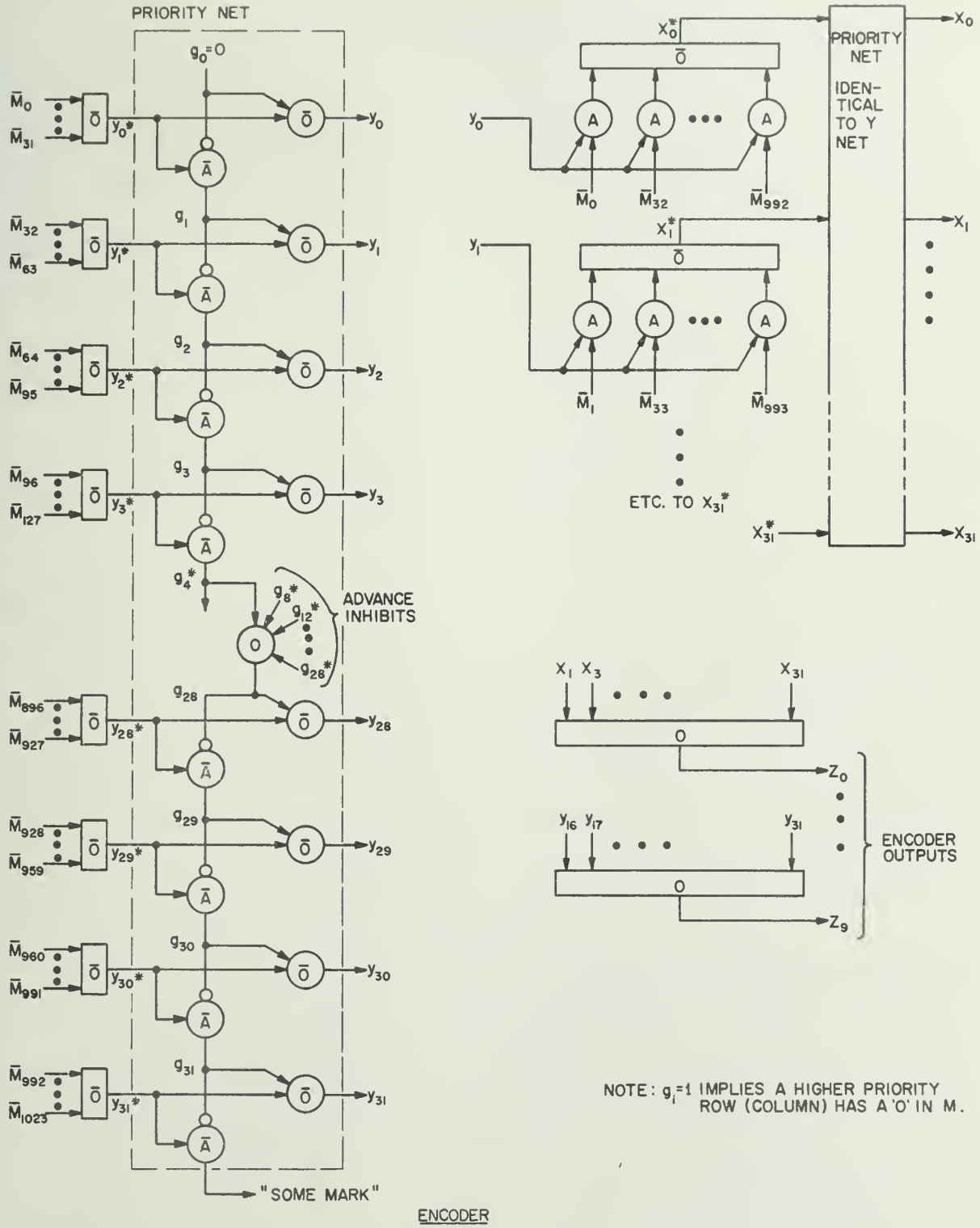
(b)

$Z=(y,x)=(1,2)=0110_2=6$ i.e., M_6

CONCEPTUAL DESCRIPTION OF ENCODER LOGIC

(a) LOCATION OF M REGISTER BITS.
(b) EXAMPLE WITH 6 BEING DESIRED ENCODING.

FIG. 4



ENCODER

FIG. 5

In the present design, the chosen segment length is four groups giving 22 operations times[11] for encoding and an inhibit driver fanout of 7 ($= \frac{\sqrt{n}}{4} - 1$).

This encoder design can be described as two-stage, the two stages being the determination of y followed, in time, by x which, of course, is a function of y. A one-stage design, such as that given by Rogers and Wolinsky [11], has a higher maximum speed of operation but requires larger driver fanouts for a given speed. Also, it is well suited to production of a 1-in-1024 indication but less desirable for full encoding.

The components required to fabricate the encoder comprise about 800 transistors and 4,000 diodes for 1024 inputs.

A plurality indicator (true if two or more zeros in M) is not included in the present design but is trivial to generate.

Transfer Memory Operations

The micro operations available in the Transfer Memory are given in Table 1. There are three species of operations: (1) Storage Access, (2) Mark Operation Only, and (3) Storage Access and Mark Operation. The word type, direction of information flow, and origin of addresses are optional as indicated. Meaningless relations are indicated by XX; "Control" means that the address is supplied from the control unit in the usual way.

The data transfer for plane words is always to and from the M register. Plane word address data is always obtained from the control unit. The encoded output address and SOME MARK signal are available as a special output data byte for use, typically, in address list compilation.

Hardware

The components required for the total Transfer Memory include 12,000 transistors and 30,000 diodes; about 60 per cent of the components occur in the

TABLE 1. TRANSFER MEMORY OPERATIONS

Operation Species	Word Type	Information Flow	Z Address Origin
Storage Access	Planar	Read/Write	XX
	Short	Read/Write	Control/Encoder
Mark Only	XX	XX	Control/Encoder
Storage Access and Mark	Short	Read/Write	Control/Encoder

plane word digit drivers, sense amplifiers and closely associated circuitry. Over 2,000 transistors of the total are involved in communication (cable driving) with the IA.

Packaging is on 620 etched wiring cards of 4" × 6" dimensions in a 17" × 25" × 65" cabinet. Maximum dc one-cycle-average power consumption is two kilowatts.

4. THE ITERATIVE ARRAY

The Iterative Array (IA) is a 1024-module parallel processor comprising nine planes of horizontal registers (see Fig. 6) interconnected to provide logical functions of vertically-related cells (cells having identical x,y coordinates), of nearest neighbors in the x and y directions, and various combinations of vertical and nearest neighbor cells.

The capability of the circuitry at each x,y position of the Iterative Array is most clearly understood in terms of the logic diagram of Fig. 7. The state of any gate must be identical at all x,y positions.

Simple Logic Functions

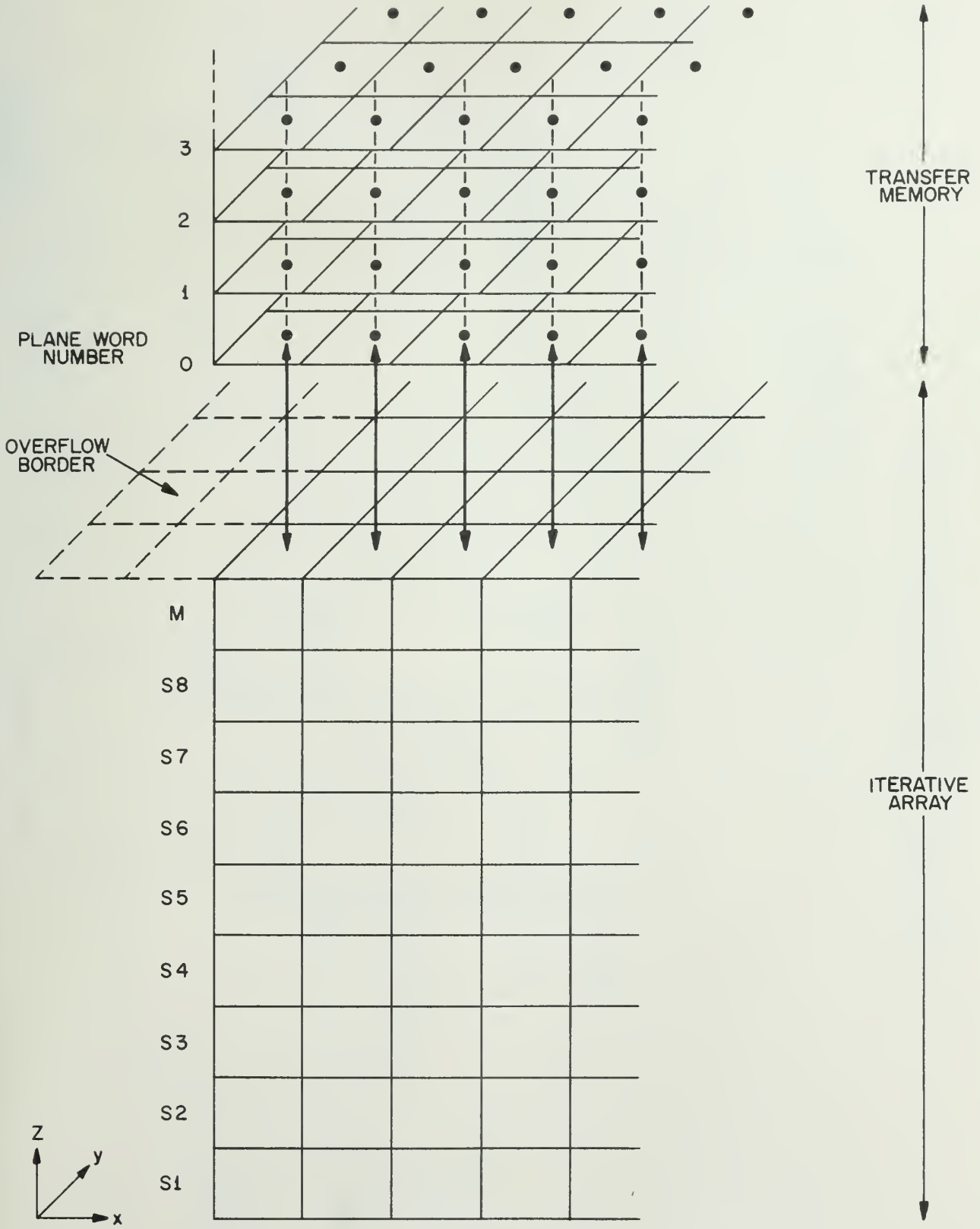
The gates controlling the OUTBUS may be operated such that any one of the following functions of the ten indicated variables is formed on the OUTBUS:

$$f_1 = M \cdot S1 \cdot S2 \cdot \ldots \cdot S8 \cdot INBUS$$
$$f_2 = \bar{f}_1$$
$$f_3 = M \vee S1 \vee S2 \vee \ldots \vee S8 \vee \overline{INBUS}$$
$$f_4 = \bar{f}_3$$

Any subset of the ten variables may be included in the arguments; the null set produces $f_1 = f_4 = 1$.

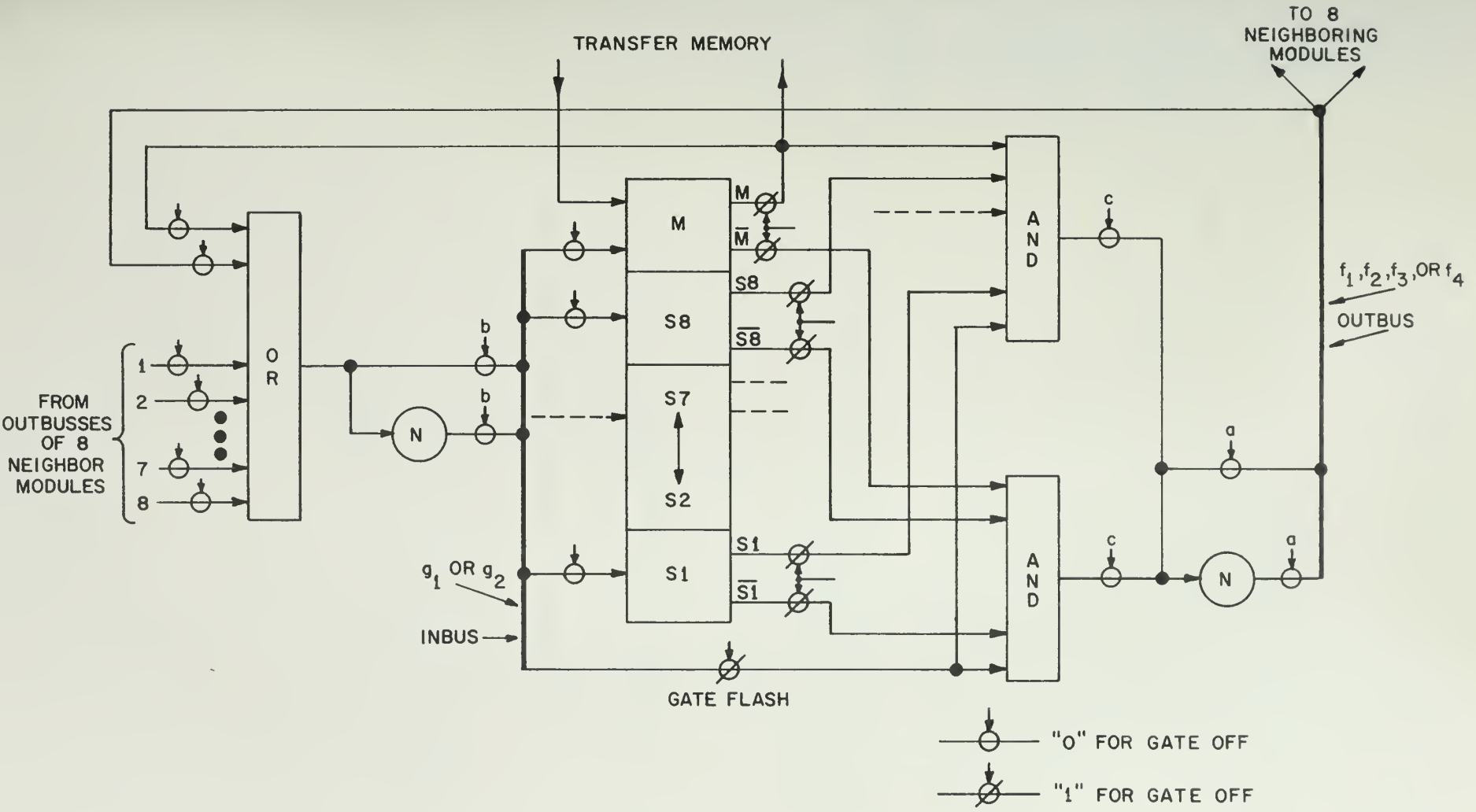
The OUTBUS function formed is made available to the eight neighboring modules as well as the originating module. Conversely, by symmetry, the OUTBUSes (1 to 8) of the nearest neighbor modules are available as inputs as well as the self OUTBUS (0) and the self M bit.

Either of the following functions may be formed on the INBUS and gated into one or more of the registers:



TRANSFER MEMORY – ITERATIVE ARRAY INTERCONNECTION

FIG. 6



LOGIC FOR ONE MODULE OF ITERATIVE ARRAY

FIG. 7

$$g_1 = M \vee (\bigvee_{i=0}^{8} OUTBUS_i)$$

$$g_2 = \bar{g}_1$$

For no arguments, $g_1 = 0$. A constraint exists on the register(s) into which the INBUS may be gated, namely, that the receiving register may not be a component of the argument of the function on the INBUS.

The formation and return of the functional value to a module constitutes a single micro-operation with a timing of 0.5 μsec including gate setup.

Stacking

The stacking operation provides an analog or base 1 counting ope ation. The subspecies, "STACK 1'S UP," may be defined by the replacement formula

$$Si' = INBUS \cdot \overline{Si} \cdot S(i - 1)$$

where i = 1, 2, ..., 8, S0 = 1 (effectively), Si' means the new value of Si and the replacement is made only if Si' is "1". In words, the value of the INBUS replaces the lowest lying zero (if any) in S1 to S8 with the register's physical relation as indicated in Fig. 6.

An example of the use of STACK is to provide a general threshold function of the number of 1's and 0's in the vicinity of each coordinate position. That is, by stacking, the function¶

$$\sum_j (\alpha_j \cdot V_j) + T$$

may be evaluated if α, V, and T are integers, the variables, V_j, are all at, or adjacent to, the x,y position under consideration, and no intermediate sum falls outside the range 0-7.

Another use of STACK is in addition-subtraction (Sec. 5).

Two variants of the STACK operation are permitted: (1) STACK 1'S UP and (2) STACK 0'S DOWN in which the highest 1 is replaced by the complement of the INBUS.

Commonly Used Microinstructions

Some simple microinstructions having general utility are introduced here and used in Sec. 5.

(1) Clear IA to 0's (1's).

This instruction sets all IA registers to 0 (or 1). By activating no gates to the input OR circuit (see Fig. 7), $g_1 = 0$ is produced on the INBUS. The INBUS-to-register gates may be activated to clear any register (S) to zero; in particular, all nine registers may be cleared. Use of $g_2 = \bar{g}_1$ obviously provides for clearing to 1's.

(2) Replicate M

This instruction causes the contents of M to be replicated in the eight storage elements of the same cell.

Form $g_1 = M$ on the INBUS and gate INBUS to S1, S2, ..., S8.

Hardware

Each of the 1024 identical modules of the Iterative Array depicted in Fig. 8 contains 39 transistors and 174 diodes used in circuits of mixed type.

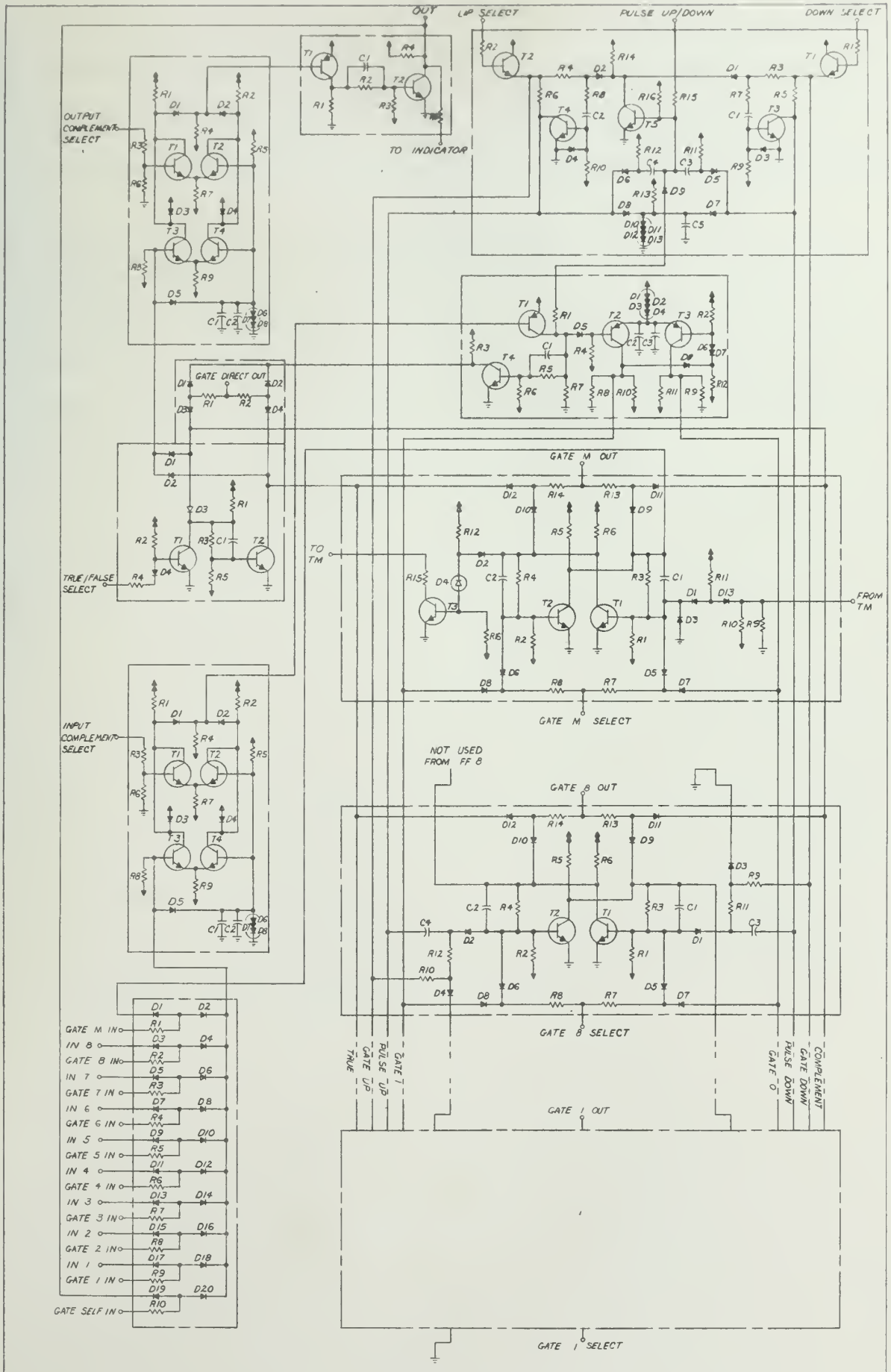


FIGURE 8. ITERATIVE ARRAY MODULE

Saturated circuitry has been used where load variations are large, notably in the flipflops and output circuits. Current switching logic has been used for its level shifting capability and passive input properties in both complementing circuits.

The IA design has been based on three major tenets. The first and most important of these has been to establish a fundamental logic in which many basic operations are simply and cheaply available. In the resulting design a total of 33 independent control inputs to each IA module provide in excess of one million useful gating combinations in each.

A second premise has been to reduce the total power supply drain. To this end resistor gating with low standby power has been used in most cases. Also it has been arranged that the high-current stacking logic circuits be selected prior to use and idle at very low power levels at other times.

The third requirement has been to distribute the load on the driver system as uniformly as possible. By this means one can avoid the extreme of an overly simple iterative module which, by its high driver requirements, unduly complicates the fanout of control to the entire array. Accordingly power gain has been introduced into the module so that a uniform fanout of 32 from all drivers may be used. Since the design has been intentionally well integrated it is only occasionally obvious where this has occurred.

5. MEMORY CONTENT SEARCH OPERATIONS

The succeeding exposition will illustrate the logical structure of the IA and TM in the role of memory content addressing and other macroinstructions. Several examples have been chosen to illustrate the nature of the calculus of various functions of memory content.

These functions are categorized by noting that examination of the memory contents, C, consists of evaluating a prescribed search function $f(C) = \{f_z\}$, where $z = 0, 1, \ldots, 1023$ and f_z is the interrogation response for memory word Z. If $f_z = f(C_z)$, that is, if f_z depends only upon the contents of memory location Z, then we say that f is a "vertical" function; if $f_z = f(C_{z*})$ with $z* \neq z$, then f is said to be a "horizontal" function; otherwise, if $f_z = f(C_z, C_{z*})$, the search is "mixed." By extension, if a majority of arguments come from words other than word Z, the function is said to be "strongly horizontal."

In most nongeometrically interpreted searches, the vertical function is of primary import whereas all three types have applications in the present case.

The following description will be couched in microinstruction terms, rather than a higher level language, in order to illustrate, most directly, the capabilities of the equipment. It should be understood that a typical user will be provided with a high level language which maps the true structure into a simplified one.

Simple Search

The object of a simple search is to identify words having a precise state in a specified group of bits or, equivalently, to evaluate a single minterm. This is a purely vertical function. After transferring the required variable planes from memory to the IA a function of up to seven variables (i.e., a 7-bit tag) can be processed in two microinstructions (at most).

The search function may be expressed as a minterm, grouped, and the complement function formed. A simple illustration will suffice. Assume the function to be evaluated is $x_1 x_2 \bar{x}_3 \bar{x}_4 x_5 = g$ (or tag 11001) and that x_i is in register Si. Then performing the following two operations in the IA

$$\overline{(S1 \cdot S2 \cdot S5)} \rightarrow S6$$

$$S6 \vee S3 \vee S4 \rightarrow M \qquad (= \bar{g})$$

leaves $\bar{g}$ in M. The zeros in M thus correspond to Z words having the desired tag which may be read out in their entirety or address listed.

Therefore, for a tag of n variables, $n \leq 7$, and k responses to interrogation, the operations are:

(1) n memory accesses for variables

(2) two (or one) IA micro-instructions

and

(3) k memory accesses (Storage Access and Mark) or address listing operations (using MARK only).

For any n, the IA operations can be done in $3 \cdot \{[n/7]\} + 2$ microinstructions, at most, where [x] is the greatest integer less than x.

For long tags, simple searches are strongly memory-access-time limited as one would expect in a serial-by-bit readout. Actually, however, not only are tags normally very short (one to three bits for the principal cases) but the percentage of IA time used for purposes other than memory content search is expected to be high. On the other hand, "full associativity," the ability to use any bit (plane word) in the search function argument, is essential.

The location of photographic transparency (brightness) contours, which may be encoded as a 3-bit gray scale function of film coordinates, is an example of the use of a simple search on photographic data.

Minimum or Maximum Search

A search for the numerical minimum (or its dual, the maximum), a vertical function, is readily performed. Consider a search for the minimum three-bit tag. Using the Stack operation a number of times equal to the weight of the tag digit under consideration, the procedure for tag bits t_0, t_1, and t_2, with weights 1, 2, and 4 respectively, is shown in the flow diagram and example of Fig. 9.

The lowest-lying S plane containing at least one zero is found by copying planes S1-S8 into the M register consecutively (or using a binary chop) and examining the "SOME MARK" signal from the encoder. The marked words may then be read out or address listed.

For tags longer than three bits, the preceding operation is performed for the most significant three bits, the result placed in M and replicated in all S registers (one microinstruction) thus writing ones in all columns which failed as possible candidates for minimum. Repetition of the stack and search operation for the remaining tag bits in groups of three or less will then eventually produce a minimum indication.[o]

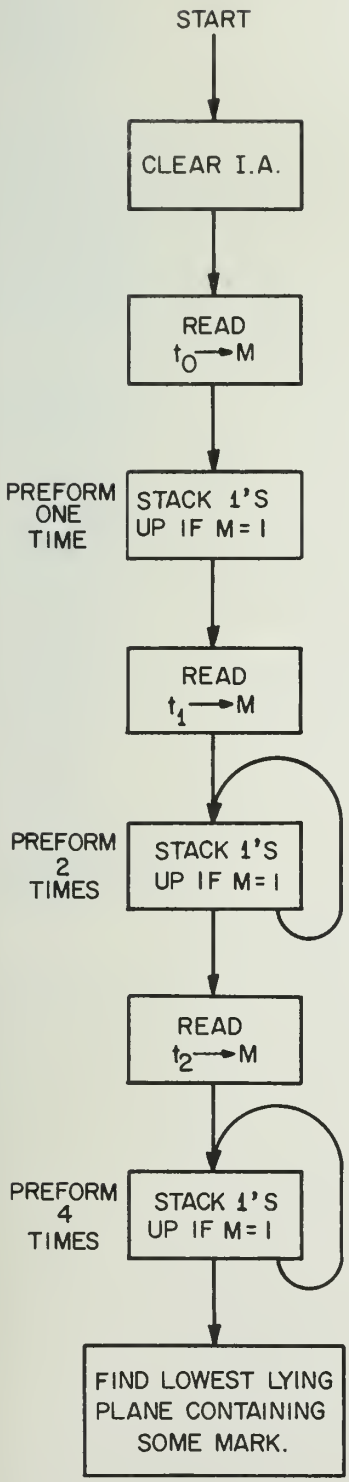
Use of STACK 1'S UP from $\overline{M}$ rather than M may be used to locate the maximum tag(s).

Approximately 4n IA micro-operations and n memory accesses are required for minimum or maximum search on n bit tags; thus the speed is 4n microseconds[√] plus the time for address listing or reading of responding words.

"Between limits" searches are also readily performed.

Addition and Subtraction

Addition in the vertical direction may be simultaneously performed on 1024 operand pairs stored in memory. The sum is produced serially by bit using



0	1	0	1	0	0	t_2
1	0	1	1	1	1	t_1
1	0	0	1	0	1	t_0

MEMORY PLANES SELECTED AS TAG DIGITS.

M						
S8	0	0	0	0	0	0
S7	0	0	0	1	0	0
S6	0	0	0	1	0	0
S5	0	0	0	1	0	0
S4	0	1	0	1	0	0
S3	1	1	0	1	0	1
S2	1	1	1	1	1	1
S1	1	1	1	1	1	1

I.A. (AT COMPLETION OF STACKING)

LOWEST LYING PLANE CONTAINING AT LEAST ONE MARK (ZERO).

MINIMUM SEARCH

FIG. 9

the following procedure which is substantially simpler than a usual half-addition sequence.

Let the ith summand digits be a_i and b_i and let c_i be the carry into the ith stage of addition. If the carry is retained in plane S1 and the sum digit, s_i, returned to storage, the sequence of operations indicated in Fig. 10 will produce the desired result.

This addition algorithm requires about seven IA steps and three memory accesses per sum digit.

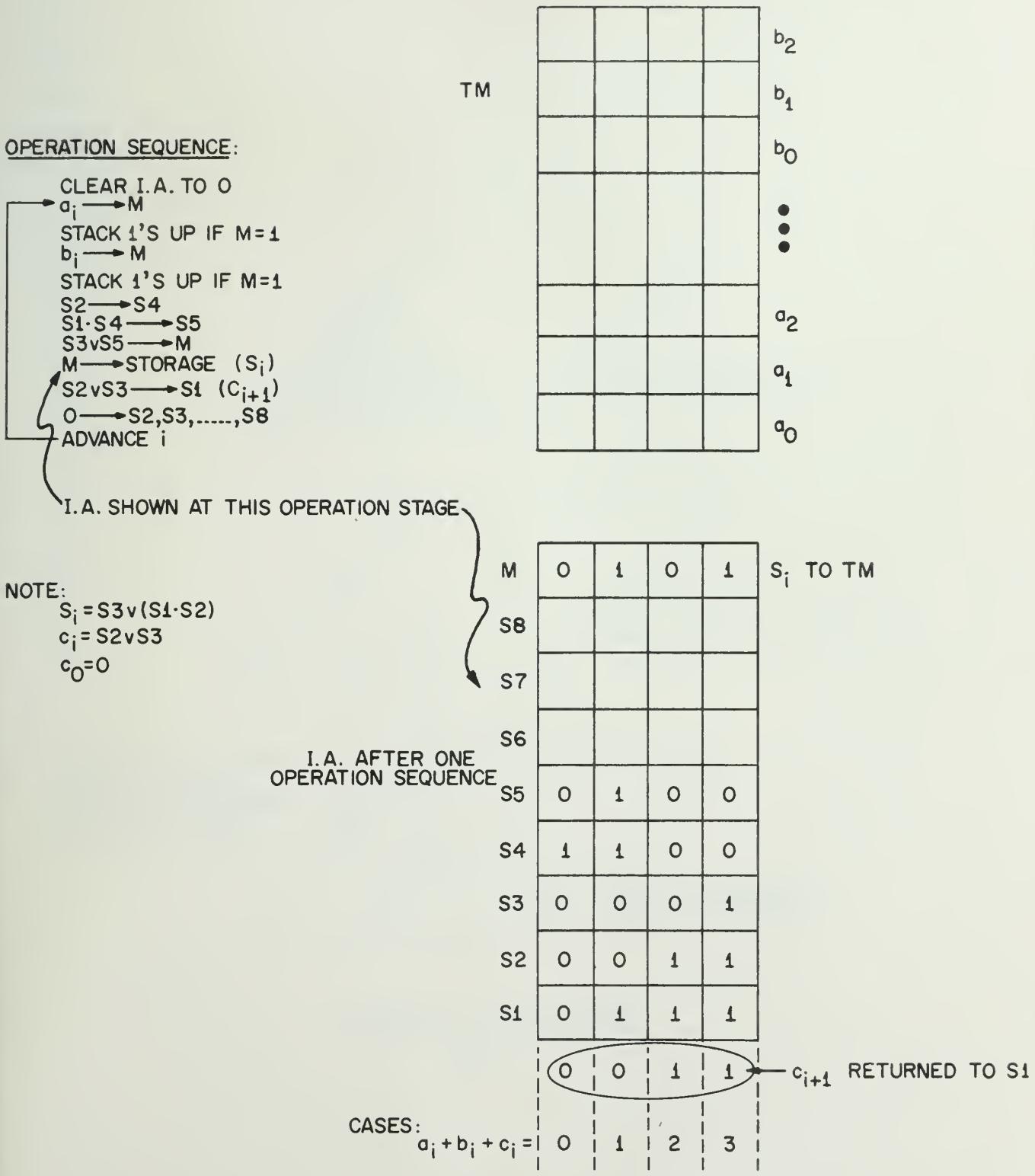
Subtraction can be implemented by a similar procedure. Addition or subtraction using an operand in a geometrically-adjacent neighbor word differs from the foregoing only in the detailed choice of gates. This fact holds generally for all preceding operations due to the similarity of the self and neighbor gates into the IA module.

Graphic Search

In the course of pattern manipulations, a digitized analog graph representing connectivity relations may be developed (see Fig. 11, plane S1). The nodes of such a graph are determined by local processing [1] and placed in another plane as, for example, in plane S2 of the same figure.

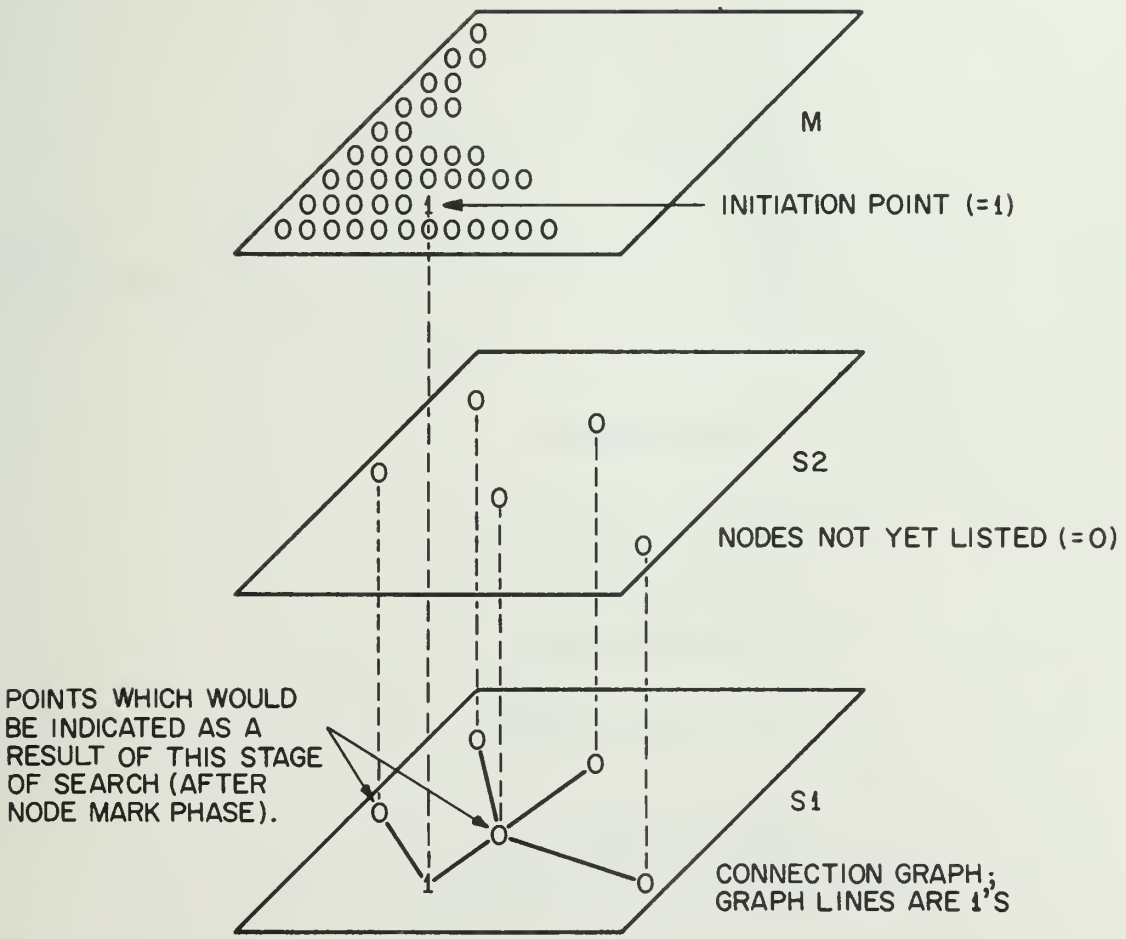
It is desired to convert the connection and node location information into an address list, or perhaps, to retrieve information from memory in graphically determined order.

The procedure consists of setting a point in the initiation plane (M) to "1," allowing the signal to propagate outward along the connection graph plane lines until the propagating signal reaches a point specified as a node by plane S2. Such points are then established as zeros in the M plane and listed. Figure 12 shows the required logical structure of the IA modules which is



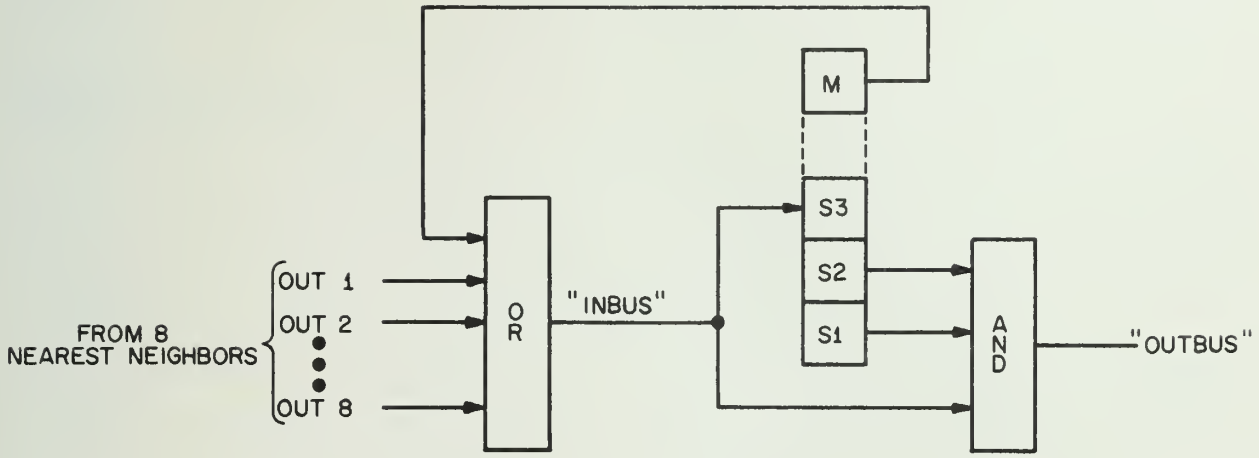
ADDITION

FIG. 10



EXAMPLE OF GRAPH SEARCH

FIG. 11



PROPAGATION PHASE

$$\left\{ \bigvee_{i=1}^{8} OUT_i \right\} \vee M \longrightarrow INBUS \longrightarrow S3$$

$$INBUS \cdot S1 \cdot S2 \longrightarrow OUTBUS$$

NODE MARK PHASE

$$\overline{S2} \longrightarrow S4$$

$$\overline{(S3 \cdot S4)} \longrightarrow M$$

(DESIRED NODES ARE 0'S IN M)

LOGIC CONNECTION DURING

GRAPHIC SEARCH (PROPAGATION PHASE)

FIG. 12

established by gate selection. The Flash gate (see Fig. 7) is active during the signal propagation mode which may consume about 3 μsec maximum.

This operation is an example of mixed search with a strongly horizontal component since the responding address may depend upon information in distant words.

Adaptive Operations

Various adaptive pattern classification operations can be programmed quite directly for the system. The correspondence to the "learning matrix" form of conditioned correlation devices is perhaps the most direct [12].

Consider, for example, 45 linear word bits of the TM to correspond to 15 pattern classes C_i with three bits representing the conductance to each of the 1024 orthogonal lines which correspond to feature sets x_i.

In the learning phase, the feature set may be loaded into the M register and temporarily stored in one word of the TM.

Using the three bits corresponding to the prescribed pattern class to transfer relevant planes to the IA, the weighting(s) of the connection(s) described by the feature class may be simultaneously altered by addition or subtraction. On the other hand, if desired, the feature class input may be used to address the pertinent Z word(s) which may be transferred to the arithmetic computer for alteration.

During the knowing phase, a presented feature set, for example, containing relatively few marks, x_j, may be used to read the conditional probability vectors from memory into the arithmetic computer. That is, the $Pr(C_i|x_j)$ vectors (Z words) for the active values of j in the feature set, may be conveniently transferred to the arithmetic calculator where the desired pattern class, C_i, may be calculated according to

$$C_i^* = \max_i \{Pr(C_i|x_j)\}$$

Upon specifying a pattern class, the corresponding feature set may be determined by a maximum search in the IA.

Although the use of this system in adaptive operations has not been investigated deeply, it is clear that the flexibility provided by logical arithmetic computations on orthogonal axes is much greater than for a one-axis computer. The high loading rate of 2 nsec/bit (96 μsec/core load) for the Transfer Memory is well adapted to searches requiring frequent core reloading, hence, the simulation of sizable matrices can reasonably be contemplated.

6. CONCLUDING REMARKS

The equipment which has been described provides an example of an iterative processor of practical design oriented toward pattern recognition problems. In particular, methods for the visual data processing of bubble-chamber negatives have been established for this processor.

The economic attainment of the necessary processing generality can be seen to rest largely upon the richly interconnected but programmable nature of the Iterative Array modules. Such arrangements are no longer impractical to program because of recent advances in the knowledge and development of high-level languages.

Stacking, by providing an easily interpreted base one or analog count, is of general use in threshold logic operations. The increased facility for decision making which it provides has been illustrated in the implementation of an addition algorithm.

The Flash procedure provides a rapid means of path-following which was illustrated for the case of conversion of a digitized analog graph to an address list or digital graph. It is also useful in generating mask words, checker-boards, and various regular patterns.

The Transfer Memory, acting in its primary role as a storage unit to the IA, necessarily includes facilities for reading/writing of 1024 bit words. In addition the realization of the MARK operation, and the reading/writing of short words (for communication with the associated arithmetic computer of more conventional design) were shown to be conveniently integrated to economic advantage.

In addition to its basic purpose, it is reasonable to expect that this processor will also be used as an experimental research tool for investigation of the application of iterative methods to other problems. This new facility

is especially significant since a computer organized along conventional lines is decidedly ill-adapted to simulation of parallel iterative processing. In particular it has been noted that the organization of the processor has a number of advantages for studying on a large scale the application of adaptive correlation techniques to various problems.

FOOTNOTES

*Received_____. This work is supported in part by Contract No. AT(11-1)-1018 of the Atomic Energy Commission.

†Digital Computer Laboratory, University of Illinois, Urbana, Illinois.

‡Fifty-four cores actually exist in the matrix, only 48 of which are typically used for information storage.

§This depends upon technology and basic circuit repertoire, of course; the numbers quoted are for typical contemporary diode-transistor logic.

||A mean time of 330 nsec for the present design.

¶This function evaluation is a macroinstruction.

°If ones occur in a three-bit group of all candidate words, then S8 would be the valid plane, containing the useful accumulated marks.

√4n micro-operations of 0.5 microseconds plus n memory accesses of two microseconds.

BIBLIOGRAPHY

[1] B. H. McCormick, "The Illinois Pattern Recognition Computer--ILLIAC III," IEEE Trans. on Elec. Computers, vol. EC-12, pp. 791-813; Dec., 1963.

[2] S. H. Unger, "A Computer Oriented Toward Spatial Problems," Proc. IRE, vol. 46, pp. 1744-1750; Dec., 1958, and "Pattern Detection and Recognition," Proc. IRE, vol. 47, pp. 1737-1752; Dec., 1959.

[3] J. H. Holland, "Iterative Circuit Computers," Proc. of WJCC, pp. 259-265; 1960.

[4] J. Gregory and R. McReynolds, "The SOLOMON Computer," IEEE Trans. on Elec. Computers, vol. EC-12, pp. 774-781; Dec., 1963.

[5] A. E. Slade and H. O. McMahon, "A Cryotron Catalog Memory System," Proc. of EJCC; Dec., 1956.

[6] J. R. Kiseda, H. E. Petersen, W. C. Seelbach, and M. Teig, "A Magnetic Associative Memory," IBM JRD, 5, No. 2, pp. 106-121; April, 1961.

[7] R. R. Seeber and A. B. Lindquist, "Associative Memory with Ordered Retrieval," IBM JRD, 6, No. 1, pp. 126-136; Jan., 1962.

[8] P. M. Davies, "A Superconductive Associative Memory," AFIPS, Proc. 1962 Spring JCC, San Francisco, Cal. (May 1-3, 1962), 79-88.

[9] R. H. Fuller, "Content-Addressable Memory Systems," Dept. of Engineering, UCLA, Report No. 63-25; June, 1963.

[10] R. Narasimhan, "Syntactic Descriptions of Pictures and Gestalt Phenomena of Visual Perception," Digital Computer Lab., Univ. of Illinois, Report No. 142; July, 1963.

[11] J. L. Rogers and A. Wolinsky, "Associative Memory Algorithms and their Cryogenic Implementation," Report No. 8670-6007-RU]000, Thompson-Ramo Wooldridge Space Technology Laboratories; Dec., 1963.

[12] H. Kazmierczak and K. Steinbuch, "Adaptive Systems in Pattern Recognition," IEEE Trans. on Elec. Computers, vol. EC-12, pp. 822-835; Dec., 1963.